M 11

```
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
```

```
**FILE**ID**INDIRECT                                             J 14
```

```
  IIIIII    NN      NN  DDDDDDDD       IIIIII    RRRRRRRR     EEEEEEEEEE    CCCCCCCC   TTTTTTTTTT
  IIIIII    NN      NN  DDDDDDDD       IIIIII    RRRRRRRR     EEEEEEEEEE    CCCCCCCC   TTTTTTTTTT
    II      NN      NN  DD      DD       II      RR      RR   EE          CC               TT
    II      NN      NN  DD      DD       II      RR      RR   EE          CC               TT
    II      NNNN    NN  DD      DD       II      RR      RR   EE          CC               TT
    II      NNNN    NN  DD      DD       II      RR      RR   EE          CC               TT
    II      NN  NN  NN  DD      DD       II      RRRRRRRR     EEEEEEEE    CC               TT
    II      NN  NN  NN  DD      DD       II      RRRRRRRR     EEEEEEEE    CC               TT
    II      NN    NNNN  DD      DD       II      RR  RR       EE          CC               TT
    II      NN    NNNN  DD      DD       II      RR  RR       EE          CC               TT
    II      NN      NN  DD      DD       II      RR    RR     EE          CC               TT          ....
    II      NN      NN  DD      DD       II      RR    RR     EE          CC               TT          ....
  IIIIII    NN      NN  DDDDDDDD       IIIIII    RR      RR   EEEEEEEEEE    CCCCCCCC       TT          ....
  IIIIII    NN      NN  DDDDDDDD       IIIIII    RR      RR   EEEEEEEEEE    CCCCCCCC       TT          ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

```
0000          .TITLE  INDIRECT - INDIRECT FILE MANIPULATION ROUTINES
0000          .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ; D. N. CUTLER 2-MAY-77
0000    ;
0000    ; INDIRECT FILE MANIPULATION ROUTINES
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-016 HWS0100         Harold Schultz  06-AUG-1984
0000    ;               Open any new indirect frame with Carriagecontrol
0000    ;               attributes.
0000    ;
0000    ;       V03-015 HWS0081         Harold Schultz  15-Jul-1984
0000    ;               When closing the current indirect frame and unstacking
0000    ;               the previous frame, set NAM block to not use the RSA and
0000    ;               ESA fields left by the indirect frame just closed. Using
0000    ;               these values cause the free dynamic memory list to become
0000    ;               corrupted. Add support for execute-only command procedures.
0000    ;
0000    ;       V03-014 HWS0080         Harold Schultz  12-Jul-1984
0000    ;               When allocating room in symbol table for resultant
0000    ;               name string, don't use constant of 256; use the value
0000    ;               of NAM$C_MAXRSS+1 rounded up to a long word boundry
0000    ;               instead. Remove the ASSUME of NAM$C_MAXRSS. Bypass
0000    ;               deallocation of unused buffer if none to deallocate.
0000    ;
0000    ;       V03-013 HWS0066         Harold Schultz  21-May-1984
0000    ;               Correct the error handling when a SYMOVF error is encountered
0000    ;               while setting up the new indirect level. Reenable password
0000    ;               masking after opening an indirect input file.
0000    ;
0000    ;       V03-012 HWS0015         Harold Schultz  21-Feb-1984
```

```
0000    ;               Check status after $FIND.
0000    ;               Initialize file spec. size fields in NAM block before reusing.
0000    ;               Deassign SYS$INPUT prior to reopening a file at a prior indirect
0000    ;               level.
0000    ;
0000    ;       V03-011 PCG0013         Peter George    12-Jan-1984
0000    ;               Fix broken branch.
0000    ;
0000    ;       V03-010 PCG0012         Peter George    17-Aug-1983
0000    ;               Correctly clear RMS F$SEARCH context.
0000    ;               Manage concealed logical name attribute using the new services.
0000    ;
0000    ;       V03-009 PCG0011         Peter George    27-May-1983
0000    ;               Fix bug in unstacking when restored command procedure
0000    ;               is already positioned to EOF.
0000    ;
0000    ;       V03-009 PCG0011         Peter George    27-May-1983
0000    ;               Fix bug in file name saving logic.
0000    ;               Fix bugs in SYS$OUTPUT processing.
0000    ;
0000    ;       V03-008 KRM0099         Karl Malik      29-Apr-1983
0000    ;               Disable password masking for network.
0000    ;
0000    ;       V03-007 PCG0010         Peter George    10-Apr-1983
0000    ;               Finish making remote open work.
0000    ;
0000    ;       V03-006 PCG0009         Peter George    22-Feb-1983
0000    ;               Add DCL$DEFINE_P1_TO_P8.
0000    ;               Clear FAB$M_SQO bit.
0000    ;               Clear FAB$V_NAM and FAB$W_IFI when performing
0000    ;               remote reopen.
0000    ;
0000    ;       V03-005 PCG0008         Peter George    28-Jan-1983
0000    ;               Remove reference to ONEXIT bit.
0000    ;
0000    ;       V03-004 PCG0007         Peter George    13-Jan-1983
0000    ;               Call SYS$OUTPUT routines.
0000    ;               Save name of command procedure.
0000    ;               Use saved file name spec to reopen command procedures
0000    ;               on remote nodes.
0000    ;
0000    ;       V03-003 PCG0006         Peter George    30-Dec-1982
0000    ;               Clear PRC_V_ONEXIT when unstacking.
0000    ;
0000    ;       V03-002 PCG0005         Peter George    28-Oct-1982
0000    ;               Fix CLRBIT typo.
0000    ;
0000    ;       V03-001 PCG0004         Peter George    15-Jul-1982
0000    ;               Allow execute-only command procedures.
0000    ;---
0000
0000    ;
0000    ; MACRO LIBRARY CALLS
0000    ;
0000
0000            PRCDEF                                  ;DEFINE PROCESS WORK AREA
0000            WRKDEF                                  ;DEFINE COMMAND WORK AREA
```

```
                                 0000                PTRDEF                          ;DEFINE RESULT PARSE DESCRIPTOR FORMAT
                                 0000                IDFDEF                          ;DEFINE INDIRECT FRAME OFFSETS
                                 0000                PRDDEF                          ;PROCESS RMS DATA
                                 0000                SYMDEF                          ;DEFINE TYPES OF SYMBOLS
                                 0000                $CLIMSGDEF                      ;DEFINE ERROR/STATUS VALUES
                                 0000                $DEVDEF                         ;DEFINE DEVICE CHARACTERISTIC BITS
                                 0000                $FABDEF                         ;DEFINE FAB OFFSETS
                                 0000                $RABDEF                         ;DEFINE RAB OFFSETS
                                 0000                $LOGDEF                         ;DEFINE LOG OFFSETS
                                 0000                $NAMDEF                         ;DEFINE NAM OFFSETS
                                 0000                $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                                 0000
                                 0000      ;
                                 0000      ; LOCAL SYMBOLS
                                 0000      ;
                                 0000
                       00000008  0000      SYMBOLS=8                                 ;MAXIMUM NUMBER OF INDIRECT FILE SYMBOLS
                                 0000
                                 0000      ;
                                 0000      ; LOCAL DATA
                                 0000      ;
                                 0000
                             00000000              .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                                 0000      INPFILE:                                   ; INPUT FILE DEFAULT NAME STRING
                    4D 4F 43 2E  0000              .ASCII  /.COM/                     ;
              54 55 50 54 55 4F  0004      OUTQUAL:.ASCII  /OUTPUT/                   ; REST OF NAME AND THE QUALIFIER
                                 000A      SYS_INPUT_NAME:
     54 55 50 4E 49 24 53 59 53 00'  000A              .ASCIC  /SYS$INPUT/            ; LOGICAL NAME FOR SYS$INPUT
                             09  000A
                                 0014      LNM$PROCESS:
53 53 45 43 4F 52 50 24 4D 4E 4C 00'  0014              .ASCIC  /LNM$PROCESS/          ; PROCESS LOGICAL NAME TABLE
                             0B  0014
           3A 30 41 4C 4E 5F 00'  0020      NLA0:   .ASCIC  /_NLA0:/               ; NULL DEVICE
                             06  0020
```

```
                        0027          .SBTTL  STACK INDIRECT FILE
                        0027  ;+
                        0027  ; DCL$STACKIND - STACK INDIRECT FILE
                        0027  ;
                        0027  ; THIS ROUTINE IS CALLED TO STACK THE CURRENT INDIRECT FILE LEVEL AND TO PARSE
                        0027  ; AND OPEN THE NEXT INDIRECT FILE.
                        0027  ;
                        0027  ; INPUTS:
                        0027  ;
                        0027  ;       IT IS ASSUMED THAT THE INDIRECT FILE PROCESSING FLAG IS SET.
                        0027  ;
                        0027  ; OUTPUTS:
                        0027  ;
                        0027  ;       THE CURRENT INDIRECT FILE SPECIFICATION IS SAVED ON THE INDIRECT FILE
                        0027  ;       STACK AND THE NEXT INDIRECT FILE IS PROCESSED.
                        0027  ;
                        0027  ;       RO LOW BIT CLEAR INDICATES INDIRECT FILE PROCESSING FAILURE.
                        0027  ;
                        0027  ;               RO = DCL$_ATLAST - INDIRECT FILE SPECIFICATION NOT LAST ITEM ON
                        0027  ;                       COMMAND LINE.
                        0027  ;               RO = DCL$_DEFOVF - ATTEMPT TO DEFINE MORE THAN EIGHT PARAMETERS.
                        0027  ;               RO = DCL$_STKOVF - INDIRECT FILE INTERNAL STACK OVERFLOW.
                        0027  ;
                        0027  ;       RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                        0027  ;
                        0027  ;               RO = DCL$_NORMAL - NORMAL COMPLETION.
                        0027  ;-
                        0027
                        0027  DCL$STACKIND::                                          ;STACK INDIRECT FILE
            0346  30    0027          BSBW    SETIND                          ;SET INDIRECT PROCESSING UP
      5E   C0 AE  9E    002A          MOVAB   -<SYMBOLS*8>(SP),SP             ;ALLOCATE SPACE FOR SYMBOL DESCRIPTO
              7E  D4    002E          CLRL    -(SP)                           ;CLEAR COUNT OF GENERATED SYMBOLS
         F4BE CA  D7    0030          DECL    WRK_L_CHARPTR(R10)              ;BACK UP TO AT SIGN
           FFC9'  30    0034  10$:    BSBW    DCL$MARK                        ;MARK CURRENT PARSE POSITION
         53   03  9A    0037          MOVZBL  #PTR_K_PARAMETR,R3              ;SET TOKEN CONTEXT FOR FILESPEC
           FFC3'  30    003A          BSBW    DCL$PROCFILE                    ;PROCESS FILE SPECIFICATION
           3C 50  E9    003D          BLBC    R0,15$                          ;IF LBC PARSE FAILURE
           FFBD'  30    0040          BSBW    DCL$SETCHAR                     ;PEEK AT NEXT CHARACTER IN INPUT BUF
         50   2F  91    0043          CMPB    #^A\/\,R0                       ;SLASH?
              37  12    0046          BNEQ    20$                             ;IF NEQ NO
           FFB5'  30    0048          BSBW    DCL$MOVTOKN                     ;MOVE TERMINATOR AND GET NEXT TOKEN
         04   51  D1    004B          CMPL    R1,#4                           ;MORE THAN MAX MATCH NAME
              03  19    004E          BLSS    13$                             ;BR IF NO
         51   04  D0    0050          MOVL    #4,R1                           ;ONLY CHECK FOR 4 CHARS
              50  DD    0053  13$:    PUSHL   R0                              ;SAVE TERMINATION CHARACTER
  AA AF   62  51  29    0055          CMPC    R1,(R2),OUTQUAL                 ;CHECK FOR VALID QUAL
              0C  13    005A          BEQL    14$                             ;BR IF OK
           50 8ED0      005C          POPL    R0                              ;RESTORE TERMINATION CHARACTER
                        005F          STATUS  IVQUAL                          ;SET ILLEGAL QUALIFIER CODE
              14  11    0066          BRB     15$                             ;
           50 8ED0      0068  14$:    POPL    R0                              ;RESTORE TERMINATION CHARACTER
         50   3D  91    006B          CMPB    #^A/=/,R0                       ;EQUAL SIGN TERMINATOR?
              C4  13    006E          BEQL    10$                             ;IF EQL YES
         50   3A  91    0070          CMPB    #^A/:/,R0                       ;COLON TERMINATOR?
              BF  13    0073          BEQL    10$                             ;IF EQL YES
                        0075          STATUS  IVVALU                          ;SET INVALID VALUE SYNTAX
            007D  31    007C  15$:    BRW     80$                             ;
```

```
                   007F     
                   007F     ;
                   007F     ; FILE SPECIFICATIONS PARSED - PARSE SYMBOL DEFINITIONS
                   007F     ;
                   007F     ; IF THE FILESPEC WAS FOLLOWED BY A SPACE, THAT SPACE MAY HAVE BEEN THROWN
                   007F     ; AWAY IF THE FIRST CHARACTER IN P1 MAKES IT INSIGNIFICANT.
                   007F     ;
   58  04 AE   9E  007F     20$:    MOVAB   4(SP),R8                        ;GET ADDRESS OF SYMBOL DESCRIPTOR ST
        FF7A'  30  0083             BSBW    DCL$SETNBLK                     ;IGNORE BLANKS AFTER FILESPEC
        FF77'  30  0086     30$:    BSBW    DCL$MARK                        ;MARK POSITION OF FIRST NON-BLANK
        FF74'  30  0089     40$:    BSBW    DCL$MOVCHAR                     ;COPY A CHARACTER FROM INPUT BUUFER
           04  E0  008C             BBS     #WRK_V_QUOTE,-                  ;LOOP IF IN A QUOTED STRING
     F8 F0 AA      008E                     WRK_Q_FLAGS(R10),40$            ;
           05  13  0091             BEQL    45$                             ;BR IF END OF LINE
      50   20  91  0093             CMPB    #^A' ',R0                       ;IS THIS A TERMINATOR
           F1  12  0096             BNEQ    40$                             ;BR IF NO - KEEP LOOKING FOR TERMINA
        FF65'  30  0098     45$:    BSBW    DCL$MARKEDTOKEN                 ;GET DESCRIPTOR OF PARAMETER
           51  D7  009B             DECL    R1                              ;REMOVE COUNT FOR TERMINATOR
           18  13  009D             BEQL    60$                             ;IF NULL STRING - NO MORE SYMBOLS
      62   22  91  009F             CMPB    #^A/"/,(R2)                     ;SYMBOL START WITH A QUOTE
           03  12  00A2             BNEQ    50$                             ;IF NO - LEAVE THE SYMBOL ALONE
        FF59'  30  00A4             BSBW    DCL$COMPRESS                    ;ELSE REMOVE THE QUOTE PAIRS
      88   51  7D  00A7     50$:    MOVQ    R1,(R8)+                        ;STORE SYMBOL DESCRIPTOR
   D8 6E   08  F3  00AA             AOBLEQ  #SYMBOLS,(SP),30$               ;ANY MORE SYMBOL DEFINITIONS ALLOWED
                   00AE             STATUS  DEFOVF                          ;SET SYMBOL DEFINITION OVERFLOW
           45  11  00B5             BRB     80$                             ;
                   00B7     
                   00B7     ;
                   00B7     ; RUN DOWN ANY IMAGE CURRENTLY RUNNING
                   00B7     ;
        BA AA  DD  00B7     60$:    PUSHL   WRK_L_RSLNXT(R10)               ;SAVE POINTER INTO WRK AREA
        FF43'  30  00BA             BSBW    DCL$RUNDOWN                     ;RUN DOWN IMAGE AND INDIRECT LEVELS
50   BA AA 8E  C3  00BD             SUBL3   (SP)+,WRK_L_RSLNXT(R10),R0      ;CALCULATE LENGTH OF STACK SHIFT
   68 AE   50  C0  00C2             ADDL    R0,<<SYMBOLS*8>+4+<9*4>>(SP)    ;RELOCATE SAVED WRK_L_RSLNXT
   6C AE   50  C0  00C6             ADDL    R0,<<SYMBOLS*8>+4+<10*4>>(SP)   ;RELOCATE SAVED WRK_L_RSLEND
   70 AE   50  C0  00CA             ADDL    R0,<<SYMBOLS*8>+4+<11*4>>(SP)   ;RELOCATE SAVED WRK_L_EXPANDPTR
   74 AE   50  C0  00CE             ADDL    R0,<<SYMBOLS*8>+4+<12*4>>(SP)   ;RELOCATE SAVED WRK_L_MARKPTR
                   00D2     
                   00D2     ;
                   00D2     ; STACK COMMAND PROCEDURE
                   00D2     ;
        68 AE  D0  00D2             MOVL    <<SYMBOLS*8>+4+<9*4>>(SP),-     ;RETRIEVE ADDRESS OF DESCRIPTORS
        BA AA      00D5                     WRK_L_RSLNXT(R10)               ;
        FF26'  30  00D7             BSBW    DCL$GETDVAL                     ;GET INPUT FILE DESCRIPTOR VALUES
      7E   51  7D  00DA             MOVQ    R1,-(SP)                        ;SAVE INPUT FILESPEC
           54  D4  00DD             CLRL    R4                              ;ASSUME NO OUTPUT FILESPEC
        FF1E'  30  00DF             BSBW    DCL$GETDVAL                     ;GET OUTPUT FILESPEC
        03 50  E9  00E2             BLBC    R0,65$                          ;IF NONE, PASS IN NULL FILESPEC
      54   51  7D  00E5             MOVQ    R1,R4                           ;SET OUTPUT FILESPEC ARGUMENT
      52   8E  7D  00E8     65$:    MOVQ    (SP)+,R2                        ;SET INPUT FILESPEC ARGUMENT
           51  D4  00EB             CLRL    R1                              ;SIGNAL ALL RMS ERRORS
         0049  30  00ED             BSBW    DCL$PUSHPROC                    ;PUSH PROCEDURE ONTO INDIRECT STACK
        09 50  E9  00F0             BLBC    R0,80$                          ;BRANCH IF ERROR DETECTED
                   00F3     
                   00F3     ;
                   00F3     ; CREATE SYMBOLS P1-P8
                   00F3     ;
           56   6E   D0  00F3   261          MOVL    (SP),R6                         ;GET NUMBER OF SYMBOL DEFINITIONS
        58   04 AE   9E  00F6   262          MOVAB   4(SP),R8                        ;GET ADDRESS OF VALUE DESCRIPTORS
                09   10  00FA   263          BSBB    DCL$DEFINE_P1_TO_P8             ;DEFINE P1 THROUGH P8
                         00FC   264
                         00FC   265 ;
                         00FC   266 ; RESTORE THE STACK AND PREPARE TO EXIT
                         00FC   267 ;
        5E   44 AE   9E  00FC   268 80$:     MOVAB   <SYMBOLS*8+4>(SP),SP            ;DEALLOCATE SYMBOL DESCRIPTOR STORAG
                50   DD  0100   269          PUSHL   R0                              ;SAVE FINAL STATUS
              024A   31  0102   270          BRW     STKXIT                          ;
```

```
                          0105          .SBTTL  DEFINE SYMBOLS P1-P8
                          0105      ;+
                          0105      ; DCL$DEFINE_P1_TO_P8 - DEFINE SYMBOLS P1-P8
                          0105      ;
                          0105      ; THIS ROUTINE IS CALLED TO DEFINE THE LOCAL SYMBOLS P1-P8.
                          0105      ;
                          0105      ; INPUTS:
                          0105      ;
                          0105      ;       R6 = NUMBER OF SYMBOLS THAT HAVE ASSIGNED VALUES
                          0105      ;       R8 = ADDRESS OF LIST OF Pn VALUE DESCRIPTORS
                          0105      ;
                          0105      ; OUTPUTS:
                          0105      ;
                          0105      ;       R1-R8 TRASHED
                          0105      ;
                          0105      ;-
                          0105
                          0105      DCL$DEFINE_P1_TO_P8::
7E   3050 8F      3C      0105              MOVZWL  #^A/P0/,-(SP)           ;CREATE PROTOTYPE OF GENERATED SYMBO
     57   08      D0      010A              MOVL    #SYMBOLS,R7             ;SET NUMBER OF SYMBOLS TO GENERATE
          51      D4      010D      10$:    CLRL    R1                      ;ASSUME NO MORE SYMBOLS DEFINED
          56      D7      010F              DECL    R6                      ;ARE THERE ANY MORE TO DEFINE
          03      19      0111              BLSS    20$                     ;BR IF NO - DEFINE AS NULL STRING
     51   88      7D      0113              MOVQ    (R8)+,R1                ;GET VALUE DESCRIPTOR
       01 AE      96      0116      20$:    INCB    1(SP)                   ;INCREMENT SYMBOL NUMBER
     53   02      D0      0119              MOVL    #2,R3                   ;SET LENGTH OF SYMBOL NAME
     54   6E      9E      011C              MOVAB   (SP),R4                 ;SET ADDRESS OF SYMBOL NAME
  55   38 AB      9E      011F              MOVAB   PRC_Q_LOCAL(R11),R5     ;GET ADDRESS OF LOCAL SYMBOL TABLE L
     50   00      D0      0123              MOVL    #SYM_K_STRING,R0        ;SET SYMBOL TYPE IS STRING
       FED7'      30      0126              BSBW    DCL$ALLOCSYM            ;ALLOCATE AND INSERT SYMBOL TABLE EN
       0A 50      E9      0129              BLBC    R0,90$                  ;IF LBC ALLOCATION FAILURE
       DE 57      F5      012C              SOBGTR  R7,10$                  ;ANY MORE SYMBOL TO PROCESS?
                          012F              STATUS  NORMAL                  ;SET NORMAL COMPLETION STATUS
          8E      D5      0136      90$:    TSTL    (SP)+                   ;RESTORE THE STACK
                  05      0138              RSB                             ;RETURN
```

```
                              0139          .SBTTL  PUSH PROCEDURE ONTO INDIRECT STACK
                              0139   ;+
                              0139   ; DCL$PUSHPROC - PUSH PROCEDURE ONTO INDIRECT STACK
                              0139   ;
                              0139   ; THIS ROUTINE IS CALLED TO INITIALIZE A NEW INDIRECT FRAME
                              0139   ; ON THE INDIRECT PROCEDURE STACK.
                              0139   ;
                              0139   ; INPUTS:
                              0139   ;
                              0139   ;       R1 = 1 IF RMS ERRORS SHOULD NOT BE SIGNALED, ELSE 0
                              0139   ;       R2/R3 = DESCRIPTOR OF INPUT FILESPEC
                              0139   ;       R4/R5 = DESCRIPTOR OF OUTPUT FILESPEC
                              0139   ;       R11 = ADDRESS OF PROCESS WORK AREA
                              0139   ;
                              0139   ; OUTPUTS:
                              0139   ;
                              0139   ;       R0 = STATUS (NOT SIGNALED)
                              0139   ;
                              0139   ;-
                              0139
                              0139   DCL$PUSHPROC::
           13FE 8F      BB    0139          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,AP>
      56   00A0 CB      D0    013D          MOVL    PRC_L_STACKPT(R11),R6    ;GET CURRENT INDIRECT STACK POINTER
        58   8C A6      9E    0142          MOVAB   -IDF_K_LENGTH(R6),R8     ;CALCULATE NEW INDIRECT STACK POINTER
      00A4 CB   58      D1    0146          CMPL    R8,PRC_L_STACKLM(R11)    ;INDIRECT STACK OVERFLOW?
                0C      1E    0148          BGEQU   2$                       ;BRANCH IF OK
                              014D          STATUS  STKOVF                   ;SET INDIRECT STACK OVERFLOW
           13FE 8F      BA    0154   80$:   POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,AP>
                        05    0158          RSB
                              0159
                              0159   ;
                              0159   ; ALLOCATE ROOM IN SYMBOL TABLE FOR RESULTANT NAME STRING PRIOR TO CLOSING
                              0159   ; CURRENT INDIRECT FRAME.
                              0159   ;
                              0159
 51    00000100 8F      D0    0159   2$:    MOVL    #<<<NAM$C_MAXRSS+1>+7>&^C<7>>,R1 ;SET MAXIMUM SIZE OF
                              0160                                             ;RESULTANT NAME STRING
               FE9D'    30    0160          BSBW    DCL$ALLDYNMEM            ;ALLOCATE ROOM IN THE SYMBOL TABLE
             09 50      E8    0163          BLBS    R0,3$                    ;BR IF NO ALLOCATION ERROR
                              0166          STATUS  SYMOVF                   ;INDICATE NO MORE ROOM IN SYM TAB
                E5      11    016D          BRB     80$                      ;JUST EXIT
          59    52      D0    016F   3$:    MOVL    R2,R9                    ;SAVE ADDRESS OF ALLOCATED BLOCK
                              0172
                              0172   ;
                              0172   ; THE NEW INDIRECT FILE FRAME IS FORMED ON THE STACK AND LINKED TO ANY
                              0172   ; PREVIOUS FRAMES.  THE STACK OVERFLOW CHECK HAS BEENY PERFORMED AT THIS POINT
                              0172   ;
                              0172
             5C AB      D6    0172          INCL    PRC_L_INDEPTH(R11)       ; SET NEW INSTACK STACK DEPTH
             7C AB      D6    0175          INCL    PRC_L_INDCLOCK(R11)      ; INCREMENT TOTAL STACKS OR UNSTACKS
      00A0 CB   58      D0    0178          MOVL    R8,PRC_L_STACKPT(R11)    ; ALLOCATE NEW STACK FRAME
      56   00BC CB      D0    017D          MOVL    PRC_L_IDFLNK(R11),R6     ; GET ADDRESS OF CURRENT INDIRECT FRAME
           68   56      D0    0182          MOVL    R6,IDF_L_LNK(R8)         ; LINK NEW FRAME INTO TOP OF
      00BC CB   68      DE    0185          MOVAL   IDF_L_LNK(R8), -         ; INDIRECT FILE FRAME LIST
                              018A                  PRC_L_IDFLNK(R11)
                              018A   ;
                              018A   ; R6 = Pointer to current stack frame
```

```
                            018A    ; R8 = Pointer to new stack frame
                            018A    ;
          5C    1C AB   D0  018A            MOVL    PRC_L_INDFAB(R11),AP    ;GET ADDRESS OF INDIRECT FAB
       10 A6    38 AB   7D  018E            MOVQ    PRC_Q_LOCAL(R11),IDF_Q_LOCAL(R6) ;SAVE LOCAL SYMBOL TABLE LISTHEAD
       18 A6    30 AB   7D  0193            MOVQ    PRC_Q_LABEL(R11),IDF_Q_LABEL(R6) ;SAVE LABEL SYMBOL TABLE LISTHEAD
       06 A6    6A AB   B0  0198            MOVW    PRC_W_ONLEVEL(R11),IDF_W_ONLEVEL(R6) ;SAVE ON ERROR LEVEL NUMBER
       08 A6    6C AB   D0  019D            MOVL    PRC_L_ONERROR(R11),IDF_L_ONERROR(R6) ;SAVE ON ERROR COMMAND TEXT
          50    38 AB   9E  01A2            MOVAB   PRC_Q_LOCAL(R11),R0     ;GET ADDRESS OF LOCAL TABLE LISTHEAD
             60    50   D0  01A6            MOVL    R0,(R0)                 ;SET ADDRESS OF LISTHEAD AS FORWARD LINK
             80    80   D0  01A9            MOVL    (R0)+,(R0)+             ;SET ADDRESS OF LISTHEAD AS BACKWARD LINK
          50    30 AB   9E  01AC            MOVAB   PRC_Q_LABEL(R11),R0     ;GET ADDRESS OF LABEL TABLE LISTHEAD
             60    50   D0  01B0            MOVL    R0,(R0)                 ;SET ADDRESS OF LISTHEAD AS FORWARD LINK
             80    80   D0  01B3            MOVL    (R0)+,(R0)+             ;SET ADDRESS OF LISTHEAD AS BACKWARD LINK
                6C AB   D4  01B6            CLRL    PRC_L_ONERROR(R11)      ;CLEAR ADDRESS OF ON ERROR COMMAND TEXT
    6A AB     0202 8F   B0  01B9            MOVW    #2@8!2,PRC_W_ONLEVEL(R11) ;RESET ON ERROR LEVEL TO ERROR
    60 A6     00B8 CB   D0  01BF            MOVL    PRC_L_ONCTLY(R11),IDF_L_ONCTLY(R6) ;SAVE ON CONTROL Y COMMAND
                   07   13  01C5            BEQL    5$                      ;BR IF THERE WAS NONE
 00B8 CB     0000'CF    9E  01C7            MOVAB   W^DCL$T_DEFONTXT,PRC_L_ONCTLY(R11) ;SET DEFUALT FOR NEXT LEVEL
       5E A8    01      B0  01CE    5$:     MOVW    #1@IDF_V_INPOPN,IDF_B_FLAG(R8) ;SET INPUT FILE OPEN FLAG
                            01D2                                            ;ASSUME FILE IS OPENED LOCALLY
             64 A8      D4  01D2            CLRL    IDF_L_SEARCHCTX(R8)     ;INITIALIZE F$SEARCH CONTEXT LIST
                            01D5
                            01D5    ;
                            01D5    ; CLOSE INPUT FILE FROM PREVIOUS INDIRECT LEVEL AND REMEMBER THE CURRENT
                            01D5    ; POSITION IN THE FILE, SO THAT ON RETURN, WE CAN RESET THE POSITION.
                            01D5    ;
          52    14 AB   D0  01D5            MOVL    PRC_L_INDINPRAB(R11),R2 ;SET CURRENT INDIRECT RAB POINTER
       08 AB       52   D1  01D9            CMPL    R2,PRC_L_INPRAB(R11)    ;IS THIS THE PRIMARY INPUT STREAM?
                   31   13  01DD            BEQL    7$                      ;BR IF YES-THAT NEVER GETS CLOSED
    1E 18 A2       1C   E1  01DF            BBC     #DEV$V_RND,RAB$L_CTX(R2),6$ ;SKIP IF NOT A DISK FILE
       58 A6       01   AE  01E4            MNEGW   #1,IDF_W_INPRFA(R6)     ;ASSUME END OF FILE
                            01E8            $FIND   RAB=(R2)                ;GET THE CURRENT RECORD POSITION (IT
     0000'8F       50   B1  01F1            CMPW    R0,#RMS$_EOF&^XFFFF     ;MAY HAVE BEEN ADVANCED BY AN INDIRECT
                   0A   13  01F6            BEQL    6$                      ;ACCESSOR SINCE OUR LAST $GET).
    58 A6       10 A2   D0  01F8            MOVL    RAB$W_RFA(R2),IDF_W_INPRFA(R6) ;SAVE RECORD POSITION IN FILE
    5C A6       14 A2   B0  01FD            MOVW    RAB$W_RFA+4(R2),IDF_W_INPRFA+4(R6)
    02 AC       04 A6   B0  0202    6$:     MOVW    IDF_W_INPIFI(R6),FAB$W_IFI(AP) ;SET INTERNAL FILE IDENTIFICATION
                            0207            $CLOSE  FAB=(AP)                ;
                            0210
                            0210    ;
                            0210    ; OPEN INPUT PROCEDURE FILE
                            0210    ;
             04 A8      B4  0210    7$:     CLRW    IDF_W_INPIFI(R8)        ;CLEAR INPUT FILE INTERNAL INDEX
             02 AC      B4  0213            CLRW    FAB$W_IFI(AP)           ;CLEAR INTERNAL FILE INDEX
                            0216
          51    04 AE   7D  0216            MOVQ    4(SP),R1                ;GET INPUT FILESPEC (R2/R3 ON ENTRY)
       34 AC       51   90  021A            MOVB    R1,FAB$B_FNS(AP)        ;SET SIZE OF FILE NAME STRING
       2C AC       52   D0  021E            MOVL    R2,FAB$L_FNA(AP)        ;SET ADDRESS OF FILE NAME STRING
       35 AC       04   90  0222            MOVB    #4,FAB$B_DNS(AP)        ;SET SIZE OF DEFAULT NAME STRING
    30 AC     FDD6 CF   9E  0226            MOVAB   INPFILE,FAB$L_DNA(AP)   ;SET ADDRESS OF DEFAULT NAME STRING
                            022C
          57    28 AC   D0  022C            MOVL    FAB$L_NAM(AP),R7        ;GET ADDRESS OF INDIRECT NAME BLOCK
                            0230
          69    FF 8F   90  0230            MOVB    #255,(R9)               ;STORE LENGTH OF BUFFER (IN SYM TAB)
       68 A8       59   D0  0234            MOVL    R9,IDF_L_FILENAME(R8)   ;STORE ADDRESS OF BUFFER (IN SYM TAB)
                            0238            ASSUME  NAM$B_RSL EQ NAM$B_RSS+1
                FF 8F   9B  0238            MOVZBW  #NAM$C_MAXRSS,-         ;SAVE THE SIZE OF THE BUFFER
```

```
              02 A7       023B                  NAM$B_RSS(R7)                   ;(NOTE, NOT THE ALLOCATED SIZE)
     04 A7    01 A9   9E  023D        MOVAB     1(R9),NAM$L_RSA(R7)             ;SAVE THE ADDRESS OF THE BUFFER
                          0242        ASSUME    NAM$B_RSL EQ NAM$B_RSS+1
              FF 8F   9B  0242        MOVZBW    #NAM$C_MAXRSS,-                 ;SET UP EXPANDED STRING TOO
              0A A7       0245                  NAM$B_ESS(R7)                   ;
              04 A7   D0  0247        MOVL      NAM$L_RSA(R7),-                 ;
              0C A7       024A                  NAM$L_ESA(R7)                   ;
                          024C
      08 A7      01   90  024C        MOVB      #NAM$M_PWD,NAM$B_NOP(R7);DISABLE PASSWORD MASKING
     16 AC    80 8F   90  0250        MOVB      #FAB$M_EXE,FAB$B_FAC(AP);SET FILE ACCESS TYPE
       000C0000 8F    D0  0255        MOVL      #FAB$M_INP!FAB$M_PPF,-  ;SET FILE OPEN OPTIONS
              04 AC       025B                  FAB$L_FOP(AP)           ;
      1E AC      04   90  025D        MOVB      #FAB$M_PRN,FAB$B_RAT(AP);SET CARRIAGE CONTROL
      1F AC      03   90  0261        MOVB      #FAB$C_VFC,FAB$B_RFM(AP);SET VERT. FORMS CONTROL
              17 AC   94  0265        CLRB      FAB$B_SHR(AP)           ;CLEAR FILE SHARING OPTIONS
         50      5C   D0  0268        MOVL      AP,R0                   ;ADDRESS OF FAB
                          026B
         51      04   D0  026B        MOVL      #4,R1                   ;ASSUME OPEN WITH ERROR REPORTING
              03 6E   E9  026E        BLBC      (SP),8$                 ;IF ERROR REPORTING DISABLED,
         51      02   C8  0271        BISL      #2,R1                   ;DO OPEN WITHOUT ERROR REPORTING
               FD89'  30  0274  8$:   BSBW      DCL$OPEN_CREATE         ;OPEN INDIRECT INPUT FILE
                          0277        CLRBIT    NAM$V_PWD,NAM$B_NOP(R7) ;UNCONDITIONALLY REENABLE PASSWORD MASKINNG
              3D 50   E9  027B        BLBC      R0,9$                   ;IF LBC OPEN FAILURE
                          027E
     04 A8    02 AC   B0  027E        MOVW      FAB$W_IFI(AP),IDF_W_INPIFI(R8) ;SAVE INPUT FILE INTERNAL INDEX
     56     00F4 CC   9E  0283        MOVAB     PRD_G_ALTINPRAB(AP),R6  ;GET ALTERNATE INPUT RAB
     18 A6    40 AC   D0  0288        MOVL      FAB$L_DEV(AP),RAB$L_CTX(R6) ;SAVE DEVICE CHARACTERISTICS
     0C A8    18 A6   D0  028D        MOVL      RAB$L_CTX(R6),IDF_L_INPRABCTX(R8) ;AND A COPY IN THE STACK FRAME
                 11   E1  0292        BBC       #NAM$V_NODE,-           ;BRANCH IF NOT A REMOTE OPEN
           04 34 A7       0294                  NAM$L_FNB(R7),10$       ;
                          0297        SETBIT    IDF_V_REMOTE,-          ;SET REMOTE OPEN FLAG
                          0297                  IDF_W_FLAG(R8)          ;
                          029B
                          029B ;
                          029B ; GET THE DEVICE NAME.  PROPAGATE CONCEALED ATTRIBUTES.
                          029B ;
                          029B  10$:  CLRBIT    IDF_V_INPCCL,IDF_B_OUTFLAGS(R8) ;CLEAR CONCEALED BIT IN IDF
                 0C   E1  029F        BBC       #NAM$V_CNCL_DEV,-               ;IS DEVICE CONCEALED?
           04 34 A7       02A1                  NAM$L_FNB(R7),11$               ;
                          02A4        SETBIT    IDF_V_INPCCL,IDF_B_OUTFLAGS(R8) ;SET CONCEALED BIT IN IDF
                          02A8        ASSUME    IDF_W_INPFID EQ IDF_T_INPDVI+16
                          02A8        ASSUME    IDF_W_INPDID EQ IDF_W_INPFID+6
      14 A7      1C   28  02A8  11$:  MOVC      #28,NAM$T_DVI(R7),-             ;COPY FILE INFORMATION
              3C A8       02AC                  IDF_T_INPDVI(R8)                ;INTO INDIRECT STACK FRAME
      3C A6      5C   D0  02AE        MOVL      AP,RAB$L_FAB(R6)                ;LINK FAB TO RAB
                          02B2        $CONNECT RAB=(R6)                         ;CONNECT TO NEW INPUT
              4A 50   E9  02BB  9$:   BLBC      R0,50$                          ;IF LBC CONNECT FAILURE
                          02BE        CLRBIT    RAB$V_PPF_IND,RAB$W_ISI(R6)     ;MAKE SURE INDIRECT FLAG IS CLEAR
      14 AB      56   D0  02C3        MOVL      R6,PRC_L_INDINPRAB(R11)         ;SET INDIRECT INPUT RAB
               0272   30  02C7        BSBW      SAV_EXE_ONLY            ;SAVE VER. FLAGS IF EXE-ONLY PROCEDURE.
                          02CA
                          02CA ;
                          02CA ; CREATE OUTPUT FILE, IF SPECIFIED
                          02CA ;
         51   0C AE   7D  02CA        MOVQ      12(SP),R1               ;GET OUTPUT FILESPEC (R4/R5 ON ENTRY)
         7E   03 A7   9A  02CE        MOVZBL    NAM$B_RSL(R7),-(SP)     ;SAVE LENGTH OF INPUT FILE NAME
         56      68   D0  02D2        MOVL      IDF_L_LNK(R8),R6        ;SET ADDRESS OF DEFAULT SYS$OUTPUT INFO
```

```
                    FD28'   30  02D5           BSBW    DCL$OPEN_OUTPUT           ;CONDITIONALLY OPEN SYS$OUTPUT
                      51  8ED0  02D8           POPL    R1                        ;RESTORE LENGTH OF INPUT FILE NAME
                   2A 50    E9  02DB           BLBC    R0,50$                    ;RETURN ANY ERRORS
                                02DE
                                02DE   ;
                                02DE   ; DEALLOCATE UNUSED BUFFER.
                                02DE   ;
              50   68 A8    D0  02DE           MOVL    IDF_L_FILENAME(R8),R0     ;GET ADDRESS OF BUFFER
                 60   51    90  02E2           MOVB    R1,(R0)                   ;SAVE FILE NAME LENGTH IN FIRST BYTE OF BUFF
                 51   08    C0  02E5           ADDL    #8,R1                     ;ROUND UP SIZE TO QUADWORD BOUNDARY(INCLUDE
                 51   07    CA  02E8           BICL    #7,R1                     ;TRUNCATE DOWN SIZE TO QUADWORD BOUNDARY
                 50   51    C0  02EB           ADDL    R1,R0                     ;CALCULATE ADDRESS OF UNUSED BUFFER
51   00000100 8F      51    C3  02EE           SUBL3   R1,#<<<NAM$C_MAXRSS+1>+7>&^C<7>>,R1 ;CALCULATE SIZE OF UNUSED BUFFER
                      03    13  02F6           BEQL    40$                       ;DON'T DEALLOCATE IF NO UNUSED BUFFER
                    FD05'   30  02F8           BSBW    DCL$DEADYNMEM             ;DEALLOCATE UNUSED BUFFER
                                02FB
                                02FB   ;
                                02FB   ; CREATE LOGICAL NAMES FOR 'INPUT' AND 'OUTPUT' AND EXIT WITH SUCCESS.
                                02FB   ;
                    FD02'   30  02FB   40$:    BSBW    DCL$CREATE_IO             ;CREATE LOGICAL NAMES FOR 'INPUT' AND 'OUTPU
                                02FE           STATUS  NORMAL                    ;
                    FE4C    31  0305           BRW     80$                       ;EXIT WITH SUCCESS
                                0308
                                0308   ;
                                0308   ; OPEN, CONNECT, OR SYMBOL ALLOCATION FAILURE
                                0308   ;
                   02 A7    B4  0308   50$:    CLRW    NAM$B_RSS(R7)             ;INVALIDATE RESULTANT STRINGS
                   0A A7    B4  030B           CLRW    NAM$B_ESS(R7)             ;INVALIDATE EXPANDED STRINGS
                      50    DD  030E           PUSHL   R0                        ;SAVE ERROR/STATUS VALUE
                    0080    30  0310           BSBW    UNSTACK                   ;UNSTACK PREVIOUS INDIRECT FILE
                      50  8ED0  0313           POPL    R0                        ;RETRIEVE ERROR/STATUS VALUE
                    FE38    31  0316           BRW     80$                       ;EXIT WITH STATUS
```

```
                      0319        .SBTTL  UNSTACK INDIRECT FILE SPECIFICATION
                      0319  ;+
                      0319  ; DCL$UNSTACK - UNSTACK INDIRECT FILE SPECIFICATION
                      0319  ;
                      0319  ; THIS ROUTINE IS CALLED TO CLOSE THE CURRENT INDIRECT FILE AND TO UNSTACK THE
                      0319  ; PREVIOUS SPECIFICATION.
                      0319  ;
                      0319  ; INPUTS:
                      0319  ;
                      0319  ;       NONE.
                      0319  ;
                      0319  ; OUTPUTS:
                      0319  ;
                      0319  ;       THE CURRENT INDIRECT FILE IS CLOSED AND ALL LOCAL SYMBOLS FOR THE LEVEL
                      0319  ;       ARE DEALLOCATED. THE PREVIOUS INDIRECT FILE IS THEN UNSTACKED AND REOPENED.
                      0319  ;
                      0319  ;       R0 LOW BIT CLEAR INDICATES UNSUCCESSFUL COMPLETION.
                      0319  ;
                      0319  ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                      0319  ;
                      0319  ;       ALL ERRORS ARE SIGNALED BEFORE RETURNING TO CALLER.
                      0319  ;-
                      0319
                      0319  DCL$UNSTACK::                                   ;UNSTACK INDIRECT FILE SPECIFICATION
                      0319          $DELLOG_S       TBLFLG=#LOG$C_PROCESS,- ;DELETE ANY USER DEFINED
                      0319                          ACMODE=#PSL$C_USER      ;LOGICAL NAMES.
           48    10   0326          BSBB    SETIND                          ;SETUP INDIRECT PROCESSING
           00'   DD   0328          PUSHL   S^#SS$_NORMAL                   ;ASSUME NORMAL COMPLETION
  13 68 AB 04    E5   032A          BBCC    #PRC_V_GOTO,PRC_W_FLAGS(R11),10$ ;IF CLR, NO GOTO IN PROGRESS
        FCCE'    30   032F          BSBW    DCL$DEALGOTO                    ;DEALLOCATE GOTO SYMBOL
                      0332          STATUS  USGOTO                          ;SET UNSATISFIED GOTO STATUS
       6E  50    D0   0339          MOVL    R0,(SP)                         ;SET COMPLETION STATUS
                      033C          ERRMSG                                  ;OUTPUT ERROR MESSAGE
        FCBE'    30   033F          BSBW    DCL$SET_STATUS                  ;GIVE ERROR HANDLER'S A CHANCE
           4F    10   0342  10$:    BSBB    UNSTACK                         ;UNSTACK TO PREVIOUS LEVEL
      F895 CA    9E   0344          MOVAB   WRK_G_INPBUF-1(R10),-           ;SET STARTING ADDRESS OF INPUT
      F48E CA         0348                  WRK_L_CHARPTR(R10)              ;BUFFER AS LAST BYTE FETCHED
      F896 CA    94   034B          CLRB    WRK_G_INPBUF(R10)               ;SET EOL AS NEXT BYTE TO FETCH
      11FF 8F    BA   034F  STKXIT: POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,AP> ;RESTORE REGISTERS
                      0353                                                  ; R0=STATUS, R1=ORIGINAL FLAGS
      BA AA    8ED0   0353          POPL    WRK_L_RSLNXT(R10)               ;RESTORE TOKEN DESCRIPTORS BACK TO
      B6 AA    8ED0   0357          POPL    WRK_L_RSLEND(R10)               ;WHERE THEY WERE WHEN WE STARTED
    F486 CA    8ED0   035B          POPL    WRK_L_EXPANDPTR(R10)            ;RESTORE EXPANSION BUFFER POINTER
    F48A CA    8ED0   0360          POPL    WRK_L_MARKPTR(R10)              ;RESTORE MARKER POINTER
                      0365          ENABLE                                  ;ENABLE CONTROL Y/C AST'S
    04 51  01    E0   0367          BBS     #WRK_V_COMMAND,R1,10$           ;BR IF COMMAND WAS SET
    F0 AA  02    AA   036B          BICW    #WRK_M_COMMAND,WRK_W_FLAGS(R10) ;CLEAR COMMAND IN PROGRESS
                 05   036F  10$:    RSB                                     ;
                      0370
                      0370  ;
                      0370  ; SETIND - SETUP INDIRECT
                      0370  ;
                      0370  ; SAVE THE NON-VOLATILE REGISTERS AND THE COMMAND WORK FLAGS, THEN SET COMMAND
                      0370  ;
                      0370  ;
           01    BA   0370  SETIND: POPR    #^M<R0>                         ;GET RETURN PC
                      0372          DISABLE                                 ;DISABLE CONTROL Y/C AST'S
```

```
                   F48A CA   DD  0378   569        PUSHL   WRK_L_MARKPTR(R10)          ;SAVE CURRENT MARKER POINTER
                   F486 CA   DD  037C   570        PUSHL   WRK_L_EXPANDPTR(R10)        ;SAVE CURRENT EXPANSION BUFFER POINTER
                     B6 AA   DD  0380   571        PUSHL   WRK_L_RSLEND(R10)           ;SAVE CURRENT ENDING TOKEN ADDRESS
                     BA AA   DD  0383   572        PUSHL   WRK_L_RSLNXT(R10)           ;SAVE CURRENT POSITION IN TOKEN ARRAY
                   11FC 8F   BB  0386   573        PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,AP> ;SAVE REGISTERS
                     F0 AA   DD  038A   574        PUSHL   WRK_W_FLAGS(R10)            ;SAVE PREVIOUS COMMAND FLAGS
                                 038D   575        SETBIT  WRK_V_COMMAND,WRK_W_FLAGS(R10) ;SET COMMAND IN PROGRESS
                        60   17  0391   576        JMP     (R0)                        ;RETURN TO CALLER
```

```
                          0393          .SBTTL  UNSTACK NEXT INDIRECT FILE
                          0393  ;---
                          0393  ;
                          0393  ; UNSTACK - UNSTACK NEXT INDIRECT FILE
                          0393  ;
                          0393  ; THIS ROUTINE IS CALLED TO CLOSE THE CURRENT INDIRECT FILE AND UNSTACK THE
                          0393  ; CONTEXT INFORMATION FOR THE PREVIOUS LEVEL INDIRECT FILE.
                          0393  ;
                          0393  ; INPUTS:
                          0393  ;
                          0393  ;       R11 = ADDRESS OF PROCESS WORK AREA
                          0393  ;
                          0393  ; OUTPUTS:
                          0393  ;
                          0393  ;       NONE
                          0393  ;
                          0393  ;       R0-R8,AP ARE DESTROYED.
                          0393  ;---
                          0393
                          0393  UNSTACK:                                        ;UNSTACK INDIRECT FILE
      5C    1C AB    D0   0393          MOVL    PRC_L_INDFAB(R11),AP            ;GET ADDRESS OF SCRATCH FAB
   58     00BC CB    D0   0397          MOVL    PRC_L_IDFLNK(R11),R8            ;GET ADDRESS OF CURRENT INDIRECT FRAME
               58    DD   039C          PUSHL   R8                              ;SAVE THAT ADDRESS
                          039E
                          039E  ;
                          039E  ; CLOSE CURRENT INPUT PROCEDURE FILE
                          039E  ;
   02 AC    04 A8    B0   039E          MOVW    IDF_W_INPIFI(R8),FAB$W_IFI(AP) ;RESTORE INTERNAL FILE INDEX
                          03A3          $CLOSE  FAB=(AP)                        ;CLOSE INDIRECT INPUT FILE
                          03AC
                          03AC  ;
                          03AC  ; DEALLOCATE LOCAL SYMBOLS AND LABELS FOR CURRENT LEVEL
                          03AC  ;
      53    38 BB    0F   03AC  10$:    REMQUE  @PRC_Q_LOCAL(R11),R3            ;REMOVE NEXT ENTRY FROM LOCAL SYMBOL TABLE
               06    1C   03B0          BVC     20$                             ;IF VC ENTRY REMOVED
      53    30 BB    0F   03B2          REMQUE  @PRC_Q_LABEL(R11),R3            ;REMOVE NEXT ENTRY FROM LOCAL LABEL TABLE
               05    1D   03B6          BVS     30$                             ;IF VS TABLE EMPTY
             FC45'   30   03B8  20$:    BSBW    DCL$DEALLOCSYM                  ;DEALLOCATE SYMBOL ENTRY
               EF    11   03BB          BRB     10$                             ;
                          03BD  ;
                          03BD  ; DEALLOCATE F$SEARCH CONTEXT BLOCKS FOR CURRENT LEVEL
                          03BD  ;
      53    64 A8    D0   03BD  30$:    MOVL    IDF_L_SEARCHCTX(R8),R3          ;GET FIRST ENTRY OFF F$SEARCH LIST
               26    13   03C1          BEQL    32$                             ;BRANCH IF NONE LEFT
   64 A8       63    D0   03C3          MOVL    (R3),IDF_L_SEARCHCTX(R8)        ;REMOVE FROM LINKED LIST
 3C A3    FC55 CF    90   03C7          MOVB    NLA0,FAB$B_FNS+8(R3)            ;SET NULL DEVICE NAME
 34 A3    FC50 CF    9E   03CD          MOVAB   NLA0+1,FAB$L_FNA+8(R3)          ;
                          03D3          $PARSE  FAB=8(R3)                       ;TERMINATE SEARCH SEQUENCE
         50    53    D0   03DD          MOVL    R3,R0                           ;SET ADDRESS OF BLOCK TO DEALLOCATE
      51    04 A0    D0   03E0          MOVL    4(R0),R1                        ;GET SIZE OF ENTRY IN BYTES
             FC19'   30   03E4          BSBW    DCL$DEADYNMEM                   ;DEALLOCATE CONTEXT BLOCK
               D4    11   03E7          BRB     30$                             ;LOOP UNTIL LIST CLEANED OUT
                          03E9
                          03E9  ;
                          03E9  ; DEALLOCATE FILE NAME STRING
                          03E9  ;
      50    68 A8    D0   03E9  32$:    MOVL    IDF_L_FILENAME(R8),R0           ;GET ADDRESS OF ASCIC FILENAME
```

```
                51    60    9A  03ED              MOVZBL  (R0),R1                         ;GET SIZE OF FILENAME
                51    08    C0  03F0              ADDL    #8,R1                           ;ROUND UP SIZE TO QUAD BOUNDARY
                51    07    CA  03F3              BICL    #7,R1                           ;TRUNCATE SIZE TO QUAD BOUNDARY
                   FC07'    30  03F6              BSBW    DCL$DEADYNMEM                   ;DEALLOCATE BUFFER
                                03F9
                                03F9        ;
                                03F9        ; RESET ON CONDITIONS BACK TO DEFAULTS
                                03F9        ;
                   FC04'    30  03F9              BSBW    DCL$ONRESET                     ;RESET ON ERROR PARAMETERS
                   FC01'    30  03FC              BSBW    DCL$ONCTLYRST                   ;AND THE ON CONTROL Y HANDLER
                                03FF        ;
                                03FF        ; CHECK IF THE FRAME JUST CLOSED WAS THE FIRST EXE-ONLY FRAME ENCOUNTERED.
                                03FF        ; IF SO, RESTORE VERIFICATION STATE FROM SAVED FLAGS.
                                03FF        ;
                    0178    30  03FF              BSBW    RES_EXE_ONLY                    ;CHECK EXE-ONLY PARAMETERS.
                                0402        ;
                                0402        ; POINT BACK TO THE PREVIOUS INDIRECT FRAME
                                0402        ;
       00BC CB      68      D0  0402              MOVL    IDF_L_LNK(R8), -                ; UNLINK FRAME FROM INDIRECT LIST
                                0407                      PRC_L_IDFLNK(R11)               ; AND RESET FRAME POINTER
  00A0 CB       74 A8       9E  0407              MOVAB   IDF_K_LENGTH(R8), -             ; REMOVE CURRENT INDIRECT FRAME FROM
                                040D                      PRC_L_STACKPT(R11)              ; STACK AND RESET STACK POINTER
                5C AB       D7  040D              DECL    PRC_L_INDEPTH(R11)              ; SET NEW INDIRECT STACK DEPTH
                7C AB       D6  0410              INCL    PRC_L_INDCLOCK(R11)             ; COUNT TOTAL STACKS OR UNSTACKS
       58     00BC CB       D0  0413              MOVL    PRC_L_IDFLNK(R11),R8            ; POINT TO PREVIOUS INDIRECT FRAME
                                0418
                                0418        ;
                                0418        ; RESTORE THE SAVED CONTEXT FROM THE PREVIOUS INDIRECT FRAME
                                0418        ;
       38 AB    10 A8       7D  0418              MOVQ    IDF_Q_LOCAL(R8),PRC_Q_LOCAL(R11) ;RESTORE LOCAL SYMBOL TABLE LISTHEA
       30 AB    18 A8       7D  041D              MOVQ    IDF_Q_LABEL(R8),PRC_Q_LABEL(R11) ;RESTORE LOCAL LABEL TABLE LISTHEAD
       6A AB    06 A8       B0  0422              MOVW    IDF_W_ONLEVEL(R8),PRC_W_ONLEVEL(R11) ;RESTORE ON ERROR LEVEL NUMBER
       6C AB    08 A8       D0  0427              MOVL    IDF_L_ONERROR(R8),PRC_L_ONERROR(R11) ;RESTORE ADDRESS OF COMMAND TEX
     00B8 CB    60 A8       D0  042C              MOVL    IDF_L_ONCTLY(R8),PRC_L_ONCTLY(R11) ;AND THE ON CONTROL T HANDLER
                                0432
                                0432        ;
                                0432        ; RE-OPEN THE INPUT PROCEDURE FILE ASSOCIATED WITH THE PREVIOUS
                                0432        ; INDIRECT FRAME.
                                0432        ;
   FFFF 8F      58 A8       B1  0432              CMPW    IDF_W_INPRFA(R8),#^XFFFF;IS THE INPUT FILE ALREADY AT EOF?
                   03       12  0438              BNEQ    35$                             ;NO, THEN BRANCH
                 00EA       31  043A              BRW     50$                             ;YES, THEN DO NOT REOPEN
                08 AB       D0  043D        35$:  MOVL    PRC_L_INPRAB(R11),-             ;ASSUME RETURNING TO LEVEL ZERO AND-
                14 AB           0440                      PRC_L_INDINPRAB(R11)            ;SET INPUT AS INDIRECT INPUT ALSO
      03 5E A8     00       E0  0442              BBS     #IDF_V_INPOPN,IDF_W_FLAG(R8),351$ ;CONTINUE IF NOT GOING TO LEVEL 0
                 008A       31  0447              BRW     371$                            ;SKIP IF GOING TO LEVEL 0
                                044A
                   02       DD  044A        351$: PUSHL   #PSL$C_SUPER                    ;PUSH ACCESS MODE
       7E     FBBB CF       9E  044C              MOVAB   SYS_INPUT_NAME+1,-(SP)          ;BUILD LOGICAL NAME DESCRIPTOR
       7E     FBB5 CF       9A  0451              MOVZBL  SYS_INPUT_NAME,-(SP)            ;
       7E     FBBB CF       9E  0456              MOVAB   LNM$PROCESS+1,-(SP)             ;BUILD TABLE NAME DESCRIPTOR
       7E     FBB5 CF       9A  045B              MOVZBL  LNM$PROCESS,-(SP)               ;
                51    5E    D0  0460              MOVL    SP,R1                           ;SAVE ADDR. OF DESCRIPTORS
                                0463              $DELLNM_S       TABNAM=(R1),-           ;DELETE SYS$INPUT
                                0463                      LOGNAM=8(R1),-
                                0463                      ACMODE=16(R1)
                5E    14    C0  0472              ADDL    #4*5,SP                         ;CLEAN STACK
```

```
                          0475
   56   00F4 CC     9E    0475              MOVAB   PRD_G_ALTINPRAB(AP),R6   ;GET THE ALTERNATE INPUT RAB
     14 AB    56    D0    047A              MOVL    R6,PRC_L_INDINPRAB(R11)  ;SET THAT IS INDIRECT INPUT RAB
           0C A8    D0    047E              MOVL    IDF_L_INPRABCTX(R8),-    ;RESTORE STACKED DEVICE CHARACTERISTICS-
           18 A6          0481                      RAB$L_CTX(R6)            ;VALUE FROM STACK FRAME
   57      28 AC    D0    0483              MOVL    FAB$L_NAM(AP),R7         ;ADDRESS OF NAME BLOCK
                          0487              ASSUME  IDF_W_INPFID EQ IDF_T_INPDVI+16
                          0487              ASSUME  IDF_W_INPDID EQ IDF_W_INPFID+6
   3C A8      1C    28    0487              MOVC    #28,IDF_T_INPDVI(R8),-   ;COPY PREVIOUS INPUT DEVICE,FILE AND-
           14 A7          048B                      NAM$T_DVI(R7)            ;DIRECTORY ID'S INTO NAME BLOCK
                          048D              ASSUME  NAM$B_DEV EQ NAM$B_NODE+1
                          048D              ASSUME  NAM$B_DIR EQ NAM$B_DEV+1
                          048D              ASSUME  NAM$B_NAME EQ NAM$B_DIR+1
                          048D              ASSUME  NAM$B_TYPE EQ NAM$B_NAME+1
                          048D              ASSUME  NAM$B_VER EQ NAM$B_TYPE+1
           38 A7    D4    048D              CLRL    NAM$B_NODE(R7)           ;INIT. FILE SPEC. SIZE FIELDS BEFORE
                          0490                                               ;REUSING NAM BLOCK.
           3C A7    B4    0490              CLRW    NAM$B_TYPE(R7)           ;
                          0493
                          0493              ASSUME  NAM$B_RSL EQ NAM$B_RSS+1
                          0493              ASSUME  NAM$B_ESL EQ NAM$B_ESS+1
           02 A7    B4    0493              CLRW    NAM$B_RSS(R7)            ;SET RESULT RESULTANT AND EXPANDED
           0A A7    B4    0496              CLRW    NAM$B_ESS(R7)            ;STRING SIZES TO NULL SO THAT THE
                          0499                                               ;RSA AND ESA WON'T BE USED.
                          0499
 16 AC     82 8F    90    0499              MOVB    #FAB$M_EXE!FAB$M_GET,FAB$B_FAC(AP) ;SET FILE ACCESS TYPE
   010C0000 8F      D0    049E              MOVL    #FAB$M_INP!FAB$M_PPF!FAB$M_NAM,- ;SET FILE OPEN OPTIONS
           04 AC          04A4                      FAB$L_FOP(AP)
           34 AC    94    04A6              CLRB    FAB$B_FNS(AP)            ;REMOVE RESIDUAL FILE NAME SIZE
           17 AC    94    04A9              CLRB    FAB$B_SHR(AP)            ;CLEAR FILE SHARING OPTIONS
           02 AC    B4    04AC              CLRW    FAB$W_IFI(AP)            ;CLEAR INPUT IFI
              01    E1    04AF              BBC     #IDF_V_REMOTE,-          ;SKIP IF NOT REMOTE ACCESS
        11 5E A8          04B1                      IDF_Q_FLAG(R8),36$       ;
                          04B4              CLRBIT  FAB$V_NAM,FAB$L_FOP(AP)  ;CLEAR OPEN BY NAM BLOCK FLAG
   50      68 A8    D0    04B9              MOVL    IDF_L_FILENAME(R8),R0    ;GET ADDRESS OF ASCIC FILENAME
   34 AC      80    90    04BD              MOVB    (R0)+,FAB$B_FNS(AP)      ;GET LENGTH OF FILE NAME
   2C AC      50    D0    04C1              MOVL    R0,FAB$L_FNA(AP)         ;GET ADDRESS OF FILE NAME
                          04C5      36$:    $OPEN   FAB=(AP)                 ;OPEN PREVIOUS INPUT
           05 50    E8    04CE              BLBS    R0,38$                   ;BRANCH IF SUCCESSFUL
            FB2C'   30    04D1      37$:    BSBW    DCL$ERRORMSG             ;REPORT ERROR MESSAGE
              3D    11    04D4      371$:   BRB     40$
                          04D6      38$:    CLRBIT  FAB$V_NAM,FAB$L_FOP(AP)  ;REMOVE OPEN BY NAME BLOCK FLAG
 04 A8     02 AC    B0    04DB              MOVW    FAB$W_IFI(AP),IDF_W_INPIFI(R8) ;SET NEW INPUT IFI
           02 A6    B4    04E0              CLRW    RAB$W_ISI(R6)            ;ZERO PREVIOUS INTERNAL SEQUENCE NUMBER
                          04E3              $CONNECT RAB=(R6)                ;CONNECT TO PREVIOUS INPUT
           E2 50    E9    04EC              BLBC    R0,37$                   ;BRANCH IF UNSUCCESSFUL
 1F 18 A6     1C    E1    04EF              BBC     #DEV$V_RND,RAB$L_CTX(R6),40$ ;SKIP IF NOT A DISK FILE
 14 A6     5C A8    B0    04F4              MOVW    IDF_W_INPRFA+4(R8),RAB$W_RFA4(R6) ;COPY RECORD FILE ADDRESS FROM
 10 A6     58 A8    D0    04F9              MOVL    IDF_W_INPRFA(R8),RAB$W_RFA(R6) ;FROM INDIRECT STACK TO RAB
              13    13    04FE              BEQL    40$                      ;BR IF PREVIOUS FILE AT TOP OF FILE
    1E A6     02    90    0500              MOVB    #RAB$C_RFA,RAB$B_RAC(R6) ;SET ACCESS MODE TO RECORD FILE ADR
                          0504              $FIND   RAB=(R6)                 ;SET TO NEXT RECORD POSITION
           C1 50    E9    050D              BLBC    R0,37$                   ;BRANCH IF UNSUCCESSFUL
                          0510              ASSUME  RAB$C_SEQ EQ 0
           1E A6    94    0510              CLRB    RAB$B_RAC(R6)            ;SET ACCESS TO SEQUENTIAL
                          0513
                          0513      ;
                          0513   ; CLOSE CURRENT OUTPUT FILE IF THE CURRENT OUTPUT FILE IS DIFFERENT
                          0513   ; FROM THE PREVIOUS LEVEL.  CREATE SYS$INPUT AND SYS$OUTPUT LOGICAL NAMES.
                          0513   ;
              58 8ED0     0513   40$:    POPL    R8                          ;GET ADDR OF JUST CLOSED IDF FRAME
    52    0094 C8   9E    0516           MOVAB   IDF_W_OUTIFI+IDF_K_LENGTH(R8),R2 ;GET ADDR OF OUTPUT FILE INFO
             FAE2'  30    051B           BSBW    DCL$RESTORE_OUTPUT          ;RESET OLD SYS$OUTPUT
    58    00BC CB   D0    051E           MOVL    PRC_L_IDFLNK(R11),R8        ;GET ADDR OF CURRENT IDF FRAME
             FADA'  30    0523           BSBW    DCL$CREATE_IO               ;CREATE 'INPUT' AND 'OUTPUT' LOGICAL NAMES
                    05    0526           RSB
                          0527
                          0527   ;
                          0527   ; DO NOT OPEN THIS INPUT FILE.  REOPEN THE NEXT ONE.
                          0527   ;
              58 8ED0     0527   50$:    POPL    R8                          ;GET ADDR OF JUST CLOSED IDF FRAME
    52    0094 C8   9E    052A           MOVAB   IDF_W_OUTIFI+IDF_K_LENGTH(R8),R2 ;GET ADDR OF OUTPUT FILE INFO
             FACE'  30    052F           BSBW    DCL$RESTORE_OUTPUT          ;RESET OLD SYS$OUTPUT
    58    00BC CB   D0    0532           MOVL    PRC_L_IDFLNK(R11),R8        ;GET ADDR OF CURRENT IDF FRAME
                58  DD    0537           PUSHL   R8                          ;SAVE THAT ADDRESS
              FE70  31    0539           BRW     10$                         ;REOPEN NEXT INPUT FILE
```

```
                        053C          .SBTTL  SAVE VERIFICATION STATE
                        053C  ;+
                        053C  ; SAV_EXE_ONLY - SAVE EXECUTE ONLY VERIFICATION STATE
                        053C  ;
                        053C  ; THIS ROUTINE CHECKS IF PROCEDURE THAT IS ABOUT TO BE EXECUTED IS THE FIRST
                        053C  ; EXECUTE-ONLY PROCEDURE ENCOUTERED SO FAR. IF SO, IT SAVES THE VERIFICATION
                        053C  ; STATES AND THE LEVEL NUMBER.
                        053C  ;
                        053C  ; INPUTS:
                        053C  ;
                        053C  ;       R11 - ADDRESS OF PROCESS WORK AREA
                        053C  ;
                        053C  ; OUTPUTS:
                        053C  ;
                        053C  ;       NONE
                        053C  ;-
                        053C
                        053C  SAV_EXE_ONLY:
       7E    56     D0  053C          MOVL    R6,-(SP)                        ;SAVE WORK REGISTER
 2B 16 AC    01     E0  053F          BBS     #FAB$V_GET,FAB$B_FAC(AP),30$    ;SKIP IF READ ACCESS
     012D CB    95      0544          TSTB    PRC_B_EXONLYL(R11)              ;FIRST ONE ENCOUNTERED?
          25    12      0548          BNEQ    30$                             ;NO, JUST SKIP IT
                        054A
          03    8A      054A          BICB    #PRC_V_SAVCMDV!PRC_V_SAVIMGV,-          ;PRESET SAVED VERIF. FLAGS
     012C CB            054C                  PRC_B_OUTFLAGS(R11)
          56    D4      054F          CLRL    R6                                      ;TURN OFF IMG. VERIF.
 05 68 AB    07     E1  0551          BBC     #PRC_V_VERIFY,PRC_W_FLAGS(R11),10$      ;SKIP IF NO VERIFY
                        0556          SETBIT  PRC_V_SAVCMDV,PRC_B_OUTFLAGS(R11)       ;SET CMD. VERIFY
05 00AF CB   07     E1  055B  10$:    BBC     #PRC_V_VERIMAGE,PRC_B_FLAGS2(R11),20$   ;SKIP IF NO VERIFY
                        0561          SETBIT  PRC_V_SAVIMGV,PRC_B_OUTFLAGS(R11)       ;SET IMG. VERIFY
      FA97'     30      0566  20$:    BSBW    DCL$SETVERIFY_IMAGE
012D CB   5C AB     90  0569          MOVB    PRC_L_INDEPTH(R11),PRC_B_EXONLYL(R11)   ;SAVE LEVEL NUMBER
       56    8E     D0  056F  30$:    MOVL    (SP)+,R6                        ;RESTORE WORK REGISTERS
                        0572          STATUS  NORMAL                          ;ALWAYS EXIT WITH SUCCESS
                05      0579          RSB                                     ;EXIT
```

```
                         057A          .SBTTL  RESTORE VERIFICATION STATE
                         057A  ;+
                         057A  ;
                         057A  ; RES_EXE_ONLY - RESTORE EXECUTE ONLY VERIFICATION STATE
                         057A  ;
                         057A  ; THIS ROUTINE CHECKS IF THE FRAME CURRENTLY BEING UNSTACKED IS THE FIRST
                         057A  ; EXE-ONLY PROCEDURE ENCOUNTERED. IF SO, IT RESTORES THE VERIFICATION STATES
                         057A  ; TO WHAT THEY WERE PRIOR TO THE EXECUTE ONLY PROCEDURE.
                         057A  ;
                         057A  ; INPUTS:
                         057A  ;
                         057A  ;       R11 = ADDRESS OF PROCESS WORK AREA
                         057A  ;
                         057A  ; OUTPUTS:
                         057A  ;
                         057A  ;       NONE
                         057A  ;+
                         057A
                         057A  RES_EXE_ONLY:
          7E   56    D0  057A          MOVL    R6,-(SP)                  ;SAVE WORK REGISTER
           5C  AB    91  057D          CMPB    PRC_L_INDEPTH(R11),-      ;IS THIS 1ST EX-ONLY LEVEL?
         012D  CB        0580                  PRC_B_EXONLYL(R11)
               27    12  0583          BNEQ    30$                       ;NO, SKIP THIS ONE
                         0585
         012D  CB    94  0585          CLRB    PRC_B_EXONLYL(R11)        ;CLEAR EXE-ONLY FLAG
                         0589          CLRBIT  PRC_V_VERIFY,PRC_W_FLAGS(R11)   ;INIT. CMD. VERIF. FLAG
   05 012C  CB   02  E1  058E          BBC     #PRC_V_SAVCMDV,PRC_B_OUTFLAGS(R11),10$  ;SKIP IF NO VERIFY
                         0594          SETBIT  PRC_V_VERIFY,PRC_W_FLAGS(R11)   ;SET CMD. VERIFICATION
                         0599
               56    D4  0599  10$:    CLRL    R6                        ;ASSUME NO IMAGE VERIFICATION
   02 012C  CB   03  E1  059B          BBC     #PRC_V_SAVIMGV,PRC_B_OUTFLAGS(R11),20$  ;SKIP IF NO VERIFY
               56    D6  05A1          INCL    R6                        ;SET FLAG TO SET IMG. VERIFICATION
                         05A3  20$:    CLRBIT  PRC_V_VERIMAGE,PRC_B_FLAGS2(R11) ;SYNC FLAG WITH LAST SET STATE
            FA54'    30  05A9          BSBW    DCL$SETVERIFY_IMAGE       ;SET/RESET IMAGE VERIFICATION
          56   8E    D0  05AC  30$:    MOVL    (SP)+,R6                  ;RESTORE WORK REGISTER
                         05AF          STATUS  NORMAL                    ;ALWAYS SIGNAL SUCCESS
                     05  05B6          RSB                               ;EXIT
                         05B7
                         05B7          .END
```

```
                                                               E 16
INDIRECT                          - INDIRECT FILE MANIPULATION ROUTINES    15-SEP-1984 23:55:59  VAX/VMS Macro V04-00       Page  20
Symbol table                                                                 4-SEP-1984 23:41:10  [DCL.SRC]INDIRECT.MAR;1          (8)

$$.TMP1                          = 00000001                    FAB$M_PPF                        = 00040000
$$.TMP2                          = 00000066                    FAB$M_PRN                        = 00000004
$$T1                             = 00000000                    FAB$V_GET                        = 00000001
CLI$_DEFOVF                      = 00038028                    FAB$V_NAM                        = 00000018
CLI$_IVQUAL                      = 00038240                    FAB$W_IFI                        = 00000002
CLI$_IVVALU                      = 00038088                    IDF_B_OUTFLAGS                     00000038
CLI$_NORMAL                      = 00030001                    IDF_C_LENGTH                       00000074
CLI$_STKO                        = 00038128                    IDF_K_LENGTH                       00000074
CLI$_SYMOV                       = 00038138                    IDF_L_FILENAME                     00000068
CLI$_USGOTO                      = 00038148                    IDF_L_INPRABCTX                    0000000C
DCL$ALLDYNMEM                      ********    X     02        IDF_L_LNK                          00000000
DCL$ALLOCSYM                       ********    X     02        IDF_L_ONCTLY                       00000060
DCL$COMPRESS                       ********    X     02        IDF_L_ONERROR                      00000008
DCL$CREATE_IO                      ********    X     02        IDF_L_OUTRABCTX                    00000024
DCL$DEADYNMEM                      ********    X     02        IDF_L_SEARCHCTX                    00000064
DCL$DEALGOTO                       ********    X     02        IDF_Q_LABEL                        00000018
DCL$DEALLOCSYM                     ********    X     02        IDF_Q_LOCAL                        00000010
DCL$DEFINE_P1_TO_P8                00000105 RG       02        IDF_T_INPDVI                       0000003C
DCL$DISABLE                        ********    X     02        IDF_T_OUTDVI                       00000028
DCL$ERRORMSG                       ********    X     02        IDF_V_INPCCL                     = 00000001
DCL$GETDVAL                        ********    X     02        IDF_V_INPOPN                     = 00000000
DCL$MARK                           ********    X     02        IDF_V_REMOTE                     = 00000001
DCL$MARKEDTOKEN                    ********    X     02        IDF_W_FLAG                         0000005E
DCL$MOVCHAR                        ********    X     02        IDF_W_INPDID                       00000052
DCL$MOVTOKN                        ********    X     02        IDF_W_INPFID                       0000004C
DCL$ONCTLYRST                      ********    X     02        IDF_W_INPIFI                       00000004
DCL$ONRESET                        ********    X     02        IDF_W_INPRFA                       00000058
DCL$OPEN_CREATE                    ********    X     02        IDF_W_ONLEVEL                      00000006
DCL$OPEN_OUTPUT                    ********    X     02        IDF_W_OUTIFI                       00000020
DCL$PROCFILE                       ********    X     02        IDF_W_OUTISI                       00000022
DCL$PUSHPROC                       00000139 RG       02        INPFILE                            00000000 R       02
DCL$RESTORE_OUTPUT                 ********    X     02        LNM$PROCESS                        00000014 R       02
DCL$RUNDOWN                        ********    X     02        LOG$C_PROCESS                    = 00000002
DCL$SETCHAR                        ********    X     02        NAM$B_DEV                        = 00000039
DCL$SETNBLK                        ********    X     02        NAM$B_DIR                        = 0000003A
DCL$SETVERIFY_IMAGE                ********    X     02        NAM$B_ESL                        = 0000000B
DCL$SET_STATUS                     ********    X     02        NAM$B_ESS                        = 0000000A
DCL$STACKIND                       00000027 RG       02        NAM$B_NAME                       = 0000003B
DCL$T_DEFONTXT                     ********    X     02        NAM$B_NODE                       = 00000038
DCL$UNSTACK                        00000319 RG       02        NAM$B_NOP                        = 00000008
DEV$V_RND                        = 0000001C                    NAM$B_RSL                        = 00000003
FAB$B_DNS                        = 00000035                    NAM$B_RSS                        = 00000002
FAB$B_FAC                        = 00000016                    NAM$B_TYPE                       = 0000003C
FAB$B_FNS                        = 00000034                    NAM$B_VER                        = 0000003D
FAB$B_RAT                        = 0000001E                    NAM$C_MAXR                       = 000000FF
FAB$B_RFM                        = 0000001F                    NAM$L_ESA                        = 0000000C
FAB$B_SHR                        = 00000017                    NAM$L_FNB                        = 00000034
FAB$C_VFC                        = 00000003                    NAM$L_RSA                        = 00000004
FAB$L_DEV                        = 00000040                    NAM$M_PWD                        = 00000001
FAB$L_DNA                        = 00000030                    NAM$T_DVI                        = 00000014
FAB$L_FNA                        = 0000002C                    NAM$V_CNCL_DEV                   = 0000000C
FAB$L_FOP                        = 00000004                    NAM$V_NODE                       = 00000011
FAB$L_NAM                        = 00000028                    NAM$V_PWD                        = 00000000
FAB$M_EXE                        = 00000080                    NLA0                               00000020 R       02
FAB$M_GET                        = 00000002                    OUTQUAL                            00000004 R       02
FAB$M_INP                        = 00080000                    PRC_B_CONTINUE                     000000F3
FAB$M_NAM                        = 01000000                    PRC_B_DEFRADIX                     000000AE
```

```
                                                                   F 16
INDIRECT                              - INDIRECT FILE MANIPULATION ROUTINES    15-SEP-1984 23:55:59  VAX/VMS Macro V04-00        Page  21
Symbol table                                                                    4-SEP-1984 23:41:10  [DCL.SRC]INDIRECT.MAR;1           (8)

PRC_B_EXMDEPMOD                  000000AD             PRC_T_OUTDVI                    0000011C
PRC_B_EXMDEPWID                  000000AC             PRC_V_GOTO                    = 00000004
PRC_B_EXONLYL                    0000012D             PRC_V_SAVCMDV                 = 00000002
PRC_B_FLAGS2                     000000AF             PRC_V_SAVIMGV                 = 00000003
PRC_B_IMGFLAG                    00000078             PRC_V_VERIFY                  = 00000007
PRC_B_OUTFLAGS                   0000012C             PRC_V_VERIMAGE                = 00000007
PRC_B_PROMPTLEN                  000000F0             PRC_W_ASTIOSB                   000000C6
PRC_C_LENGTH                     00000534             PRC_W_ASTRETN                   000000C8
PRC_G_COMMANDS                   00000133             PRC_W_ASTSTATUS                 000000C4
PRC_G_PROMPT                     000000F4             PRC_W_ATTMBX                    0000007A
PRC_K_LENGTH                     00000534             PRC_W_FLAGS                     00000068
PRC_L_CURRKEY                    00000048             PRC_W_INPCHAN                   00000064
PRC_L_EXMDEPADR                  000000A8             PRC_W_ONLEVEL                   0000006A
PRC_L_EXTARG                     00000094             PRC_W_OUTIFI                    00000114
PRC_L_EXTBLK                     0000008C             PRC_W_OUTISI                    00000116
PRC_L_EXTCOD                     0000009C             PRC_W_OUTMBXCHN                 000000CA
PRC_L_EXTHND                     00000090             PRC_W_OUTMBXREF                 000000CE
PRC_L_EXTPRM                     00000098             PRC_W_OUTMBXSIZ                 000000CC
PRC_L_IDFLNK                     000000BC             PRC_W_PMPTCTRL                  000000F1
PRC_L_IMGACTSTS                  00000080             PRC_W_WAITIOSB                  00000066
PRC_L_INDCLOCK                   0000007C             PRD_C_LENGTH                    00000214
PRC_L_INDEPTH                    0000005C             PRD_C_XLENGTH                   00000244
PRC_L_INDFAB                     0000001C             PRD_G_ALTINPRAB                 000000F4
PRC_L_INDINPRAB                  00000014             PRD_G_ALTOUTRAB                 00000138
PRC_L_INDOUTRAB                  00000018             PRD_G_FAB                       00000000
PRC_L_INPRAB                     00000008             PRD_G_INPRAB                    000000B0
PRC_L_LASTKEY                    0000004C             PRD_G_NAM                       00000050
PRC_L_LSTSTATUS                  000000B0             PRD_G_OUTRAB                    0000017C
PRC_L_ONCTLY                     000000B8             PRD_G_TRMLIST                   000001E4
PRC_L_ONERROR                    0000006C             PRD_G_XABTRM                    000001C0
PRC_L_OUTOFBAND                  000000B4             PRD_K_LENGTH                    00000214
PRC_L_OUTRAB                     0000000C             PRD_K_XLENGTH                   00000244
PRC_L_OUTRABCTX                  00000118             PRD_T_OUTDVI                    00000214
PRC_L_PPFLIST                    00000070             PRD_T_OUTFNM                    00000230
PRC_L_RECALLPTR                  0000012F             PRD_W_OUTDID                    0000022A
PRC_L_RESTART                    00000058             PRD_W_OUTFID                    00000224
PRC_L_SAVAP                      00000000             PSL$C_SUPER                   = 00000002
PRC_L_SAVFP                      00000004             PSL$C_USER                    = 00000003
PRC_L_SEVERITY                   00000050             PTR_B_LEVEL                     00000004
PRC_L_SPWN                       000000C0             PTR_B_NUMBER                    00000005
PRC_L_STACKLM                    000000A4             PTR_B_PARMCNT                   00000006
PRC_L_STACKPT                    000000A0             PTR_B_VALUE                     00000000
PRC_L_STATUS                     00000054             PTR_C_LENGTH                    0000000C
PRC_L_STS                        00000084             PTR_K_LENGTH                    0000000C
PRC_L_STV                        00000088             PTR_K_PARAMETR                = 00000003
PRC_L_SYMBOL                     00000060             PTR_L_DESCR                     00000000
PRC_L_TMBX                       00000074             PTR_L_ENTITY                    00000008
PRC_L_TRMLIST                    00000010             RAB$B_RAC                     = 0000001E
PRC_Q_ALLOCREG                   00000020             RAB$C_RFA                     = 00000002
PRC_Q_COMMAND                    000000E0             RAB$C_SEQ                     = 00000000
PRC_Q_FLUSHTIME                  000000D0             RAB$L_CTX                     = 00000018
PRC_Q_GLOBAL                     00000028             RAB$L_FAB                     = 0000003C
PRC_Q_IMAGENAME                  000000D8             RAB$V_PPF_IND                 = 0000000E
PRC_Q_KEYPAD                     00000040             RAB$W_ISI                     = 00000002
PRC_Q_LABEL                      00000030             RAB$W_RFA                     = 00000010
PRC_Q_LOCAL                      00000038             RAB$W_RFA4                    = 00000014
PRC_Q_SAVEPRIV                   000000E8             RES_EXE_ONLY                    0000057A R      02
```

```
RMS$_EOF                         ********    X     02        WRK_L_VERB                         FFFFFFBE
SAV_EXE_ONLY                     0000053C R        02        WRK_M_COMMAND                    = 00000002
SETIND                           00000370 R        02        WRK_V_COMMAND                    = 00000001
SS$_NORMAL                       ********    X     02        WRK_V_QUOTE                      = 00000004
STKXIT                           0000034F R        02        WRK_W_FLAGS                        FFFFFFF0
SYMBOLS                        = 00000008                    WRK_W_FLAGS2                       FFFFFFF2
SYM_B_FLAGS                      0000000B                    WRK_W_IMGCHAN                      FFFFFFEE
SYM_B_NONUNIQUE                  0000000B                    WRK_W_PMPTLEN                      FFFFF99E
SYM_B_TYPE                       0000000A                    _$$_                             = 000000EF
SYM_K_STRING                   = 00000000
SYM_L_BL                         00000004
SYM_L_FL                         00000000
SYM_T_SYMBOL                     0000000C
SYM_W_SIZE                       00000008
SYS$CLOSE                        ********   GX     02
SYS$CONNECT                      ********   GX     02
SYS$DELLNM                       ********   GX     02
SYS$DELLOG                       ********   GX     02
SYS$FIND                         ********   GX     02
SYS$OPEN                         ********   GX     02
SYS$PARSE                        ********   GX     02
SYS_INPUT_NAME                   0000000A R        02
UNSTACK                          00000393 R        02
WRK_B_CMDOPT                     FFFFFFC3
WRK_B_MAXPARM                    FFFFFFD0
WRK_B_MINPARM                    FFFFFFD1
WRK_B_PARMCNT                    FFFFFFCE
WRK_B_PARMSUM                    FFFFFFCF
WRK_B_RECALLCNT                  FFFFFFC5
WRK_B_VALLEV                     FFFFFFC4
WRK_B_VERBTYP                    FFFFFFC2
WRK_C_LENGTH                     FFFFF486
WRK_G_BUFFER                     FFFFF492
WRK_G_INPBUF                     FFFFF896
WRK_G_RESULT                     FFFFF9B6
WRK_K_LENGTH                     FFFFF486
WRK_L_CHARPTR                    FFFFF48E
WRK_L_DISALLOW                   FFFFFFE6
WRK_L_ERRORRTN                   FFFFF9AE
WRK_L_EXPANDPTR                  FFFFF486
WRK_L_IMAGE                      FFFFFFE2
WRK_L_MARKPTR                    FFFFF48A
WRK_L_PAROUT                     FFFFFFD2
WRK_L_PMPTADDR                   FFFFF9A2
WRK_L_PROMPTRTN                  FFFFF9A6
WRK_L_PROPTR                     FFFFFFC6
WRK_L_QUABLK                     FFFFFFCA
WRK_L_READRTN                    FFFFF9AA
WRK_L_RECALLPTR                  FFFFFFEA
WRK_L_RSLEND                     FFFFFFB6
WRK_L_RSLNXT                     FFFFFFBA
WRK_L_SAVAP                      FFFFFFF8
WRK_L_SAVFP                      FFFFFFFC
WRK_L_SAVSP                      FFFFFFF4
WRK_L_SIGNALRTN                  FFFFFFD6
WRK_L_SPECRTN                    FFFFF9B2
WRK_L_TAB_VEC                    FFFFFFDE
```

```
                                           +-----------------+
                                           ! Psect synopsis !
                                           +-----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC     USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            FFFFFFFC  (    0.)  01 (  1.)  NOPIC     USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
DCL$ZCODE                        000005B7  ( 1463.)  02 (  2.)  NOPIC     USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE

                                         +------------------------+
                                         ! Performance indicators !
                                         +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                     9    00:00:00.08    00:00:02.40
Command processing                83    00:00:00.67    00:00:05.41
Pass 1                           355    00:00:15.13    00:00:44.00
Symbol table sort                  0    00:00:01.61    00:00:03.90
Pass 2                           157    00:00:03.15    00:00:10.87
Symbol table output               34    00:00:00.27    00:00:01.15
Psect synopsis output              2    00:00:00.02    00:00:00.02
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals             640    00:00:20.93    00:01:07.75

The working set limit was 1500 pages.
76452 bytes (150 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1153 non-local and 47 local symbols.
843 source lines were read in Pass 1, producing 18 object records in Pass 2.
62 pages of virtual memory were used to define 43 macros.

                                        +----------------------------+
                                        ! Macro library statistics !
                                        +----------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SYSLIB]SYSBLDMLB.MLB;1                    0
_$255$DUA28:[DCL.OBJ]DCL.MLB;1                        14
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     19
TOTALS (all libraries)                                34

1417 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:INDIRECT/OBJ=OBJ$:INDIRECT MSRC$:INDIRECT/UPDATE=(ENH$:INDIRECT)+EXECML$/LIB+LIB$:DCL/LIB+SYS$LIBRARY:SYSBLDMLB/LIB
```

0070 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

GETKEYNAM
LIS

GOTO
LIS

EXPRESS
LIS

HANDLE
LIS

IMAGECTRL
LIS

INITIAL
LIS

INDIRECT
LIS

FILECMDS
LIS

IF
LIS

IMAGEXECT
LIS